International Journal of Computer Applications (0975 - 8887)
Volume 127 - No.5, October 2015

A Compact Routing based Mapping System for the Locator/ID Separation Protocol (LISP)

A. M. Anisul Huq
Faculty Member, Department of CS,
American International University - Bangladesh (AIUB),
Dhaka - 1213, Bangladesh

Hannu Flinck
Research Maanger, Technology and Innovation,
Nokia Networks,
Espoo, Finland

ABSTRACT

The Internet and its associated global routing tables are growing at an alarming rate. Will the current routing infrastructure be able to scale itself to sustain such growth? Over the past several years, many efforts have been made to resolve this important issue. This paper offers a novel solution to this serious problem by proposing and presenting an experimental mapping system called Compact Routing based Mapping (CRM). The idea here is to combine the perceived benefits of both Compact Routing and Locator/Identifier Separation Protocol (LISP). In CRM, the critical functions that affect the scalability of the routing system are grounded to the theory of Compact Routing; so that we might overcome the shortcomings of LISP-ALT. We mitigated Compact Routing's presumption of a static network by reusing LISP's registration messages and choosing landmarks dynamically based on their capability to aggregate. The key objective of this paper is to provide proof of concept, to give first-hand experience regarding the complicacies that arise with the actual development of such a mapping system. Our work also includes a comprehensive comparison between CRM and LISP-ALT. The results suggest that, CRM would be feasible in the current Internet if deployed and it would be far less expensive than LISP-ALT.

Keywords

LISP-ALT, Compact Routing, BGP, Scalability, Aggregation

1. INTRODUCTION

As the Internet experienced an explosive growth during the last three decades, its routing system has encountered numerous challenges brought about by the unprecedented scale of the system. More recently, the Internet routing architecture confronted two new daunting challenges: firstly, the IPv4 address space became exhausted which in turn, is leading to the wide spread deployment of IPv6. Secondly, the emerging mobile access to Internet from billions of hand-held devices. The latter is further driving the demands for IPv6 to be rolled out and yet the sheer size of the IPv6 address space presents a great scaling concern for the routing tables (RTs).

According to the suggestions of Internet Research Task Force (IRTF), partitioning the address space into two may provide a way out. Here, an address space will be used for hosts residing in the edges networks; while another separate address space will be utilized for routing across the core network. One proposal for implementing this suggestion is the Locator/Identifier Separation Protocol (LISP) [2]. It is a simple IP-over-UDP tunneling protocol aimed at giving a network layer support for partitioning the address space. This separation of host addresses from the ones used for routing will require the use of a mapping between the two.

In this paper, we present the prototype of a mapping system that we refer to as, "Compact routing based mapping system for the locator identifier separation protocol (LISP)" or "CRM". Unlike LISP-ALT, we did not take the liberty of imagining a "highly aggregatable" address space. Our system's aggregation is not based on any administratively pre-allocated IP address space, rather through learning about the network reachability. It will deal with the "orphan" addresses (i.e. non-aggregatable addresses) either by delegating or by generating virtual prefixes. While designing CRM, we made a conscious effort so that it utilizes BGP in a minimal fashion and at the same time, the critical functions that affect the scalability of the routing system are grounded to the theory of compact routing. Our major contributions are summarized as follows:

—CRM is a novel idea; because in it, we effectively "fused" ideas from both Compact routing and LISP to come up with a mapping system that has the best of both worlds.

—Our key objective was to provide proof of concept. We proved that, such a system is very much feasible when we delivered the prototype of CRM, the first of its kind. The idea here was to look closely at the existing routing scalability issues and design/implement such a system that mitigates these problems.

—CRM was constructed as a "loosely" integrated mapping system. In other words, it did not require any modifications to the network stack and it is completely modular from the underlying system (e.g. OS kernel). This in turn means that, CRM has minimal dependencies; it is an independent mapping system that can be made compatible with any system (i.e. routing software, OS, topology discovery protocol etc.) by making minimal changes to its interface.

—Our ultimate aim was to come up with a mapping system that limits the role of BGP; so that, later on we can retire it and replace it with a simpler topology discovery protocol. To achieve this, CRM used BGP functionalities (e.g. attributes, path selection etc.) in the least possible way. We only used BGP's community attribute to disseminate the changes in "network state".

—The results and subsequent analysis has shown that, CRM is feasible for deployment in the current Internet. Furthermore, our

comparison has proven CRM to be far cheaper than LISP-ALT in terms of complexity.

The structure of the paper is as follows: background knowledge is presented in Section 2. On the subsequent section of background proposed schemes of software implementation is discussed. In the fourth section results are discussed.Cost Profile, possible Optimizations and Comparisons are shown in section 5. Before conclusion and future work implementation analysis of the proposed system is discussed.

2. BACKGROUND

A large number of proposals decouple the core (popularly known as locators) from the end node (popularly known as identifiers) functionality within IP addresses. Such a separation gives the ability to add IPv6 devices as end node without any change in the core IPv4 network. This idea is advantageous for host mobility, multihoming, renumbering, etc. We have chosen the Locator/ID Separation Protocol (LISP) [3] that separates the core from the end node. LISP reduces forwarding state and provides multiple paths while remaining compatible with today's Internet. Though LISP solves many of the scalability problems of BGP, it has its own limitations. Its biggest drawback is its assumption of "highly aggregatable" address space that do not match with reality. Such presumption of LISP may result in larger routing tables (RTs). The solution to this shortcoming is in compact routing which allows developing routing algorithms to meet the limits on RT size, stretch, overhead, etc. Compact routing has its own weakness as it is static in nature. LISP can help to make compact routing fit for today's dynamic networking environment. As our proposed system (CRM) is based on combination of compact routing and LISP, we will provide their short descriptions here.

2.1 Compact Routing

A routing protocol can be judged as compact if the address and header size it uses grows logarithmically, the size of the RT grows sub-linearly as new nodes are added and the path stretch is bounded by a constant. Although many papers (e.g. [1]) have dealt with universal stretch-3 compact routing scheme, *TZ* [20] came up with most improved version and is therefore at the center of our attention. Here, the term "universal" refers to the fact that, it can provide routing on any graph type (e.g. grid, tree, power-law) and "stretch" is defined as the ratio between routing path's length/cost and minimum length/cost path. Additionally, this compact routing scheme introduces the concept of Landmarks (LMs); which are actually globally known nodes. In *TZ's* [20] scheme, RT sizes do not exceed O($\sqrt{n}$) (i.e. number of LMs + a nodes neighborhood size $\leq$ O($\sqrt{n}$)). This reduction in RT state is achieved by changing the LM selection scheme to an iterative process. Here initially, an initial set of LMs, set *A*, is selected randomly with a uniform probability from all nodes. Once these LMs are selected, each node in the graph is assigned the LM closest to it. Unlike *Cowen* [1], a node's neighborhood (cluster in *TZ* terminology) is no longer its *k* nearest nodes, it is instead, formed with only those nodes that are closer to the node than to the closest LM [14]. Formally,

$$C(w) = \{v \in V \mid \delta(w, v) < \delta(A, v)\} \quad (1)$$

Equation 1 states that a node *v* is in *w*'s cluster (i.e. *C(w))* if *w* is closer to *v* than *v* is to its LM. Here, *V* is the set of all nodes in the graph, $\delta(w, v)$ is the distance between nodes *w* and *v*, $\delta(A, v) = min\{\delta(u, v) \mid u \in A\}$. If this node's cluster exceeds a specified limit, then that node is considered a candidate for becoming a LM in the next iteration of the LM selection. This process continues until all nodes have a cluster size not exceeding the limit.

To route using the *TZ* scheme, one must know the destination's name, the destination's LM, and the port that LM uses to route traffic towards the destination (this part of the "label" signifies the node's cluster or neighborhood); these three components make up the "label"/address of a node. Routers within the network make routing decisions based entirely on these labels.If the label of node *d* is *(d, a, c)* then, *d* is the destination, *a* is *d*'s landmark, and the next hop for traffic to *d* is node *c*. There is a good chance that, routing via a LM and then onward to the destination will increase the distance travelled by a packet. However, there is a limit imposed on this due to the nature of routing procedure; no *TZ* path is ever more than 3 times the shortest path between *a* and *b* (i.e. worst possible path stretch is 3). This is proved mathematically using the triangle inequality and symmetry by *TZ* [20].

2.2 Locator/ID separation protocol (LISP)

As the name indicates, the Locator/Identifier Separation Protocol (LISP) [3,18], separates the identifier and the locator roles of IP addresses and thus introducing two independent address spaces. The Endpoint IDentifier space (EID) identifies end-systems and consists of IP addresses that are only locally routable. The Routing LOCator space (RLOC) locates EIDs in the Internet topology and consists of IP addresses which are globally routable. RLOCs are handled by routers in the core Internet like it is today, maintaining routes so that packets can be forwarded between any routers. On the other hand, stub domains use EIDs, and since they are only locally routable, routers in the core Internet do not need to maintain routes towards these EIDs. The main goals of this separation are to reduce BGP RT size and the BGP churn. [1] In order to enable the communication among EIDs of different domains, LISP tunnels packets in the core Internet from the RLOC of the source EID to the RLOC of the destination EID. [2] When a packet has to be sent from a source EID to a destination EID, the sender initially creates a standard IP packet, using EIDs as source and destination addresses, that is forwarded to a border router of the source domain for tunneling. The border router, also called *Ingress Tunneling Router (ITR)*, performs a lookup (locally or through a distributed system called a Mapping System) for obtaining a mapping binding between the destination EID to its RLOC, which is the border router of the destination domain, termed as *Egress Tunneling Router ETR*. Once the mapping has been obtained, the ITR encapsulates the packet using RLOCs as source and destination IP addresses. The encapsulated packet is then forwarded as usual towards the ETR . Upon reception of the packet, the ETR decapsulates it and then delivers the original packet to the destination EID [19].

2.3 LISP: Mapping System

As explained above, ITRs acquire mappings binding EIDs to a set of RLOCs for ongoing communications via a mapping system that is a key element of LISP. So far, several mapping systems have been proposed for LISP [15]. However, only two have been deployed: LISP Alternative Topology (LISP+ALT [5]) and LISP Delegated Database Tree (LISP-DDT [7]).

[1]Routing updates can be seen as a measure of routing instability, and are often referred to as churn.

[2]Note that several RLOCs can be associated to a given EID.

LISP Alternative Topology (LISP+ALT) was the initial mapping system for LISP and it relied on a BGP overlay [5]. In LISP+ALT, ETRs store mappings they are authoritative for. Such an overlay is constructed by connecting ETRs together via tunnels. This (ETRs) overlay is called the Alternative Logical Topology (ALT) where routers are called ALT routers. Any ALT router maintains a BGP session with its neighbor and announces the EID prefixes it is authoritative for, making the EIDs routable in the ALT. It must be noted that, BGP is only used to build the ALT, not to announce mappings. To get a mapping, an ITR sends a *Map-Request* for the EID on the ALT topology. The source address for the *Map-Request* is the ITR RLOC and the destination the EID. The *Map-Request* eventually reaches an originator ETR for the EID prefix that matches the destination EID. This ETR resolves the EID and sends a *Map-Reply* directly to the ITR RLOC. *Map-Replies* are not sent through ALT [19].

We choose to compare CRM with LISP Alternative Topology (LISP-ALT)[6], as it has been experimented and widely deployed by Cisco. LISP-ALT's intention is to solve BGPs scalability problem by widely reusing BGP and assuming a "highly aggregatable" host address space. By "highly aggregatable" address space, we mean that, the IP address assignment process will be along network topological lines [8]. In our opinion, such a presumption is a glaring limitation of LISP-ALT. Such an address space will eventually erode to meet the real life needs (e.g. multihoming, traffic-engineering etc.) and lead to a "not so" aggregatable address space; ultimately resulting in the increased size growth of RTs. Our previous experience also shows that, BGP does not cope well with the rapid growth of RTs. Subsequently, the original "optimally" allocated address space will start to weigh down the network or in extreme cases may fail altogether. [3] [4]

We have refrained ourselves from comparing CRM with LISP-DDT, because it came into existence at the time when CRM was at its final stages; making it impossible for us to perform any credible comparison between them.

2.3.1 OpenLISP and its importance. OpenLISP [12] is the only open-source implementation for FreeBSD, based on the LISP draft (version 07) by *Iannone et al.* [11]. It has a "dirty slate approach" (i.e. evolutionary approach). OpenLISP provides a new socket based solution, called Mapping Sockets, to interact with the Control plane (i.e. mapping system) [17]. Mapping sockets make OpenLISP an open and flexible solution, enabling us to pair up CRM (deployed in the control plane) with it.

3. SOFTWARE IMPLEMENTATION

A major part of our work was aimed at implementing a prototype (i.e. CRM) that mainly performed the tasks of a LM with minimal dependency on BGP. Another objective was to build a working prototype that has least possible dependency over any kernel/system specific data structures; so that portability and modularity can be achieved.

The whole implementation took place inside multiple instances of Ubuntu 10.10 (Maverick Meerkat) that were executed inside VirtualBox virtual machines. For our development, we mostly used standard C libraries (GCCversion 4.4.5), adhering to the standards of POSIX.1.

In terms of functionality, the implementation can be divided into four major components:

(1) UDP client (or "xTR Extension"),
(2) UDP server,
(3) TCP client and
(4) TCP server.

It should be noted that, UDP client (that implements "xTR Extension") is not a part of the LM (though it is a part of CRM). The LM consists of TCP client, TCP server and UDP server. The following subsections provide concise descriptions of CRM's individual components.

3.1 UDP Client (or "xTR Extension")

Like any other mapping system for LISP, in CRM also, an xTR (when functioning as an authoritative ETR) sends a Map-Register message to a LM to declare the presence of an EID-prefix that it owns as well as the RLOCs that should be used for exchanging subsequent Map-Request and Map-Reply messages. In other words, these registration requests contain all the EID-to-RLOC mappings owned by that xTR; i.e., all the EID-numbered networks that are connected to that particular xTR's site.

Our UDP Client's (or "xTR Extension's") processing involves 5 steps:

(1) Read from the input file that holds the mappings between EID-prefix and RLOC into a linked-list,
(2) Process the input so that it can be put inside Map-Register messages,
(3) Send the Map-Register messages to the server,
(4) Read back the server's response i.e. Map-notify packet, and lastly,
(5) Determine whether Map-Register messages are needed to be resent to a delegated LM.

We initially built the Map-Register and the Map-Notify messages according to the LISP specification [2] and OpenLISP [12] standard. Later on, we extended their capabilities by adding couple of custom defined elements that fit the needs of CRM. These extensions will be explained momentarily.

Each of the constructed Map-Register messages consists of multiple records. Each record primarily holds the mapping between an EID-prefix and its RLOC. For the purposes of aggregation and delegation each Map-Register record also has an element called *eid_mask_len* that holds the mask length of the EID-prefix. The Boolean variable *is_delegated* is an extended attribute that determines whether the EID-prefix present in that message can be delegated or not. If the subsequent Map-Notify message indicates that the EID-prefix residing in the Map-Register message can be delegated, only then, this variable is set to one. The RLOC address is stored in another char array called *locator[LOCATOR_SIZE]*.

The Map-Notify message works as an acknowledgement for the Map-Register message. In accordance with the LISP specification [2], certain values from the Map-Register message for which the acknowledgement is made, is copied into the Map-Notify message, before sending it back to the UDP client. In order to fulfill CRMs objective, we added two custom defined elements, namely,

[3] Abuse Issues and IP Addresses. See: `http://www.iana.org/abuse/faq.html`
[4] Lisp Archive.See: `http://answerpot.com/showthread.php?1552501-EID%20Allocation%20/%20ALT%20Base/Page2`

server_ip_address[EID_PREFIX_SIZE], which is a char array containing the IP address of the TCP server part of an LM. This IP address is stored in the DB, enabling the TCP client to connect with the "correct" TCP server. Second one is the Boolean variable *is_delegated*. Its purpose is the same as in a Map-Register message. Because certain variables are copied from a Map-Register message into a Map-Notify message, the UDP client gets to know which of its Map-Register record can be delegated (actually the EID-prefix inside a Map-Register record is delegated). Based on this information (through Map-Notify message's *is_delegated* element), the UDP client then can send those Map-Register messages again to the delegated LM.

3.2 UDP Server (for Aggregation)

In addition to "servicing" UDP clients, the UDP server is responsible for performing the following major tasks, namely,

Natural Aggregation,

Delegation,

Virtual Prefix generation, and lastly

Advertise BGP's community attribute through Quagga.

3.2.1 Natural Aggregation. Route aggregation summarizes routes so that, there are fewer routes to advertise across the Internet. In CRM, the IP addresses that need to be aggregated are EID-prefixes extracted from Map-register messages. The goal is to only advertise their (i.e. EID-prefixes) SUPERNET out to the world. Deducing the SUPERNET actually means to find the best possible parent address from the extracted EID-prefixes. To do so, we have taken the following steps:

(i) Insert the extracted EID-prefixes into the nodes of a Linked-List. Assign an ANCESTOR_FLAG to each of the nodes and UNSET it by default.

(ii) For every EID-prefix, traverse the Linked-List and try to detect possible parent. If a parent is found then the ANCESTOR_FLAG is SET in the EID-prefix for which the parent is located.

(iii) At the end, the EID-prefix that is the best possible parent will have its ANCESTOR_FLAG as UNSET. Additionally, "orphan" EID-prefixes will also have their ANCESTOR_FLAGs as UNSET.

For the crucial second step, we traverse Linked-List for each node and compare between two IP addresses (i.e. EID-prefixes) to determine if one is the parent of the other.

3.2.2 Delegation. In our context, the term "delegation" means that, there is another LM that is advertising a "more aggregated" EID-prefix. The concept of delegation is entirely our novel idea and is better explained in the upcoming section .

3.2.3 Virtual Prefix generation. For the sake of "better" aggregation, the address space can be partitioned into large prefixes; i.e. larger than any aggregatable prefix in use today. These prefixes are called virtual prefixes (VP). In other words, a VP is a prefix used to aggregate its contained regular prefixes [13, 15, 16].

As said earlier, when a new EID-prefix registers into CRM, we need to decide whether an existing LM can aggregate the new EID-prefix, or if some other LM is advertising a more aggregated prefix, or if we need to enforce EID-prefix aggregation through instantiating a virtual EID-prefix. According to [4], the choice depends on the "compactness" of the system that can be calculated from the number of announced unique EID-prefixes and LMs. If the number of LMs is less than square root of the number of announced identifiers then the system is considered compact and can grow. But when the number of LMs approaches the maximum allowed in the TZ scheme [13], a virtual EID-prefix needs to be instantiated.

The critical factor that determines the possible success or failure of finding a VP is its chosen length. In CRM, choosing a VP of length /16 (labeled it as MAXGROUP within the implementation) proved to be sufficient for our purposes; as the number of EID-prefixes that we had worked on were limited.

CRM generated its VP by observing the following steps:

(i) Convert each of the aggregated and orphan EID-prefixes into binary.

(ii) Starting from MSB, extract substring from each of these EID-prefix of length MAXGROUP.

(iii) For every extracted substring (from EID-prefix), compare it with all the other substrings. If both substrings are deemed equals then DO NOTHING and move on to the next substring for further comparison.

(iv) If substrings are NOT equal then QUIT. Because unequal substrings mean that, we have failed to generate a suitable VP.

3.2.4 Hash value generation and Advertisement through Quagga. The whole process of generating a hash value by the MD5 message digest algorithm is performed through an on open source "off-the-shelf" software developed by RSA Data Security Inc. After the necessary aggregations are performed, we concatenated the routes (which is the input of the MD5 Hash function) with only those EID prefixes that did not have any ancestor. This way, only when the aggregation changes; the input to the MD5 Hash function changes and subsequently, the change in the network's state is advertised through BGP's community attribute. To put it simply, a change in network's state means there was a change in the aggregation.

For performing successful BGP advertisement, we used the BGP daemon (i.e. *bgpd*) of an open source routing software called Quagga version 0.99.4.

3.3 TCP Client (for EID Topology Discovery)

Like a "usual" TCP client, our system does not start off by creating a connection with the TCP server. Rather, every 5 seconds the client machine checks the BGP announcements received through Quagga. However, this TCP client does not know beforehand which network is advertised by the LM. To obtain this information, we have used the *popen()* function (i.e. process pipes) in *open_mode* (set to *"r"*) to execute the Quagga command *#vtysh -c "show ip bgp"*; which gives us all the entries in the BGP routing table. Afterwards, a suitable regular expression is used to extract all the IP addresses from this output. For each extracted IP address we execute, *#vtysh -c "show ip bgp IP_address"* command. Only a network that is advertised by a LM would have an *aggregator* and *community* value in the output. In our test network, the command *#vtysh -c "show ip bgp 192.168.87.0"* had produced 15.10.30.1 as *aggregator* and *2310:0* as *community* value. This means that, 15.10.30.1 is the LM's ID. These acquired *community* and *aggregator* attributes might be absent from the DB. If this is the case, then we *INSERT* this new information. Or, there might be a pre-existing *community* value for that LM. In that case, we just compare the newly acquired *community* value with its immediate previous instance for that particular LM. In both scenarios, the TCP server will be connected in the usual manner so that the client can get an updated view of the

network state. This updated view is stored in the client sides EIDList table. It must be noted that, the TCP client learns the server's address by inquiring the ServerAddress table.

3.4 TCP Server

For our purposes, a "classic" multi-client TCP server was sufficient. We employed I/O multiplexing capabilities so that, the TCP server can use both the listening socket (port number 9877) and its clients' connection sockets at the same time. In other words, the TCP server can deal with multiple clients by waiting for a request on many open sockets at the same time. To achieve this, we implemented *select()* function in our TCP server.

While serving the TCP client, this server carries out the SQL *SELECT* command on the EIDList table for a particular LM. Then we processed the extracted contents so that it can be sent to the requesting client as a char array. The *write()* system call is used to write this newly created char array in the client's socket.

4. DATABASE DESIGN

We have used MySQL (Ver 14.14 Distrib 5.1.49, for debian-linux-gnu (i686) using readline 6.1) as the DBMS for its Open-Source qualities. Within MySQL, we have created a DB called *LMID-structure* for CRM. MySQL's InnoDB storage engine is used to achieve transaction capabilities. This enables MySQL to take care of any concurrency issues that might arise. In the following figure, we show the ER diagram of the entire DB residing in CRM.

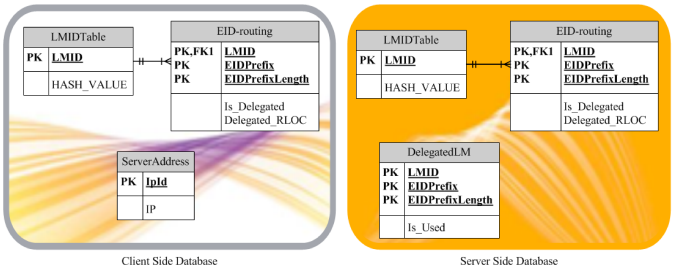

Fig. 1. complete view of the entire database residing in CRM. [9]

5. RESULTS

Before starting to analyze the results, we would like to provide a conceptual overview of CRM. Like, most mapping systems designed to operate between two distinct namespaces (e.g. LISP-ALT, LISP-DHT etc.), CRM also creates an overlay routing; where two routing systems "sit" on top of each other. One deals with EIDs, while the other with RLOCs. Here, the EID based routing is termed as the "overlay" and it involves the use of two data structures, namely, *"EID-routing table"* and *"EID-forwarding table"*. At the bottom, we have Quagga, which advertises RLOC through BGP.

As CRM is designed to provide a remedy for the routing scalability problem, we have utilized Quagga only for RLOC routing; so that its routing facilities are used in the least possible way. We shall now examine the "test" topology as shown in fig. 2.
This figure 2 shows how the architecture is implemented and deployed based on the description of the prior section 3. The whole system is deployed through three Virtual Machines (VMs). Both

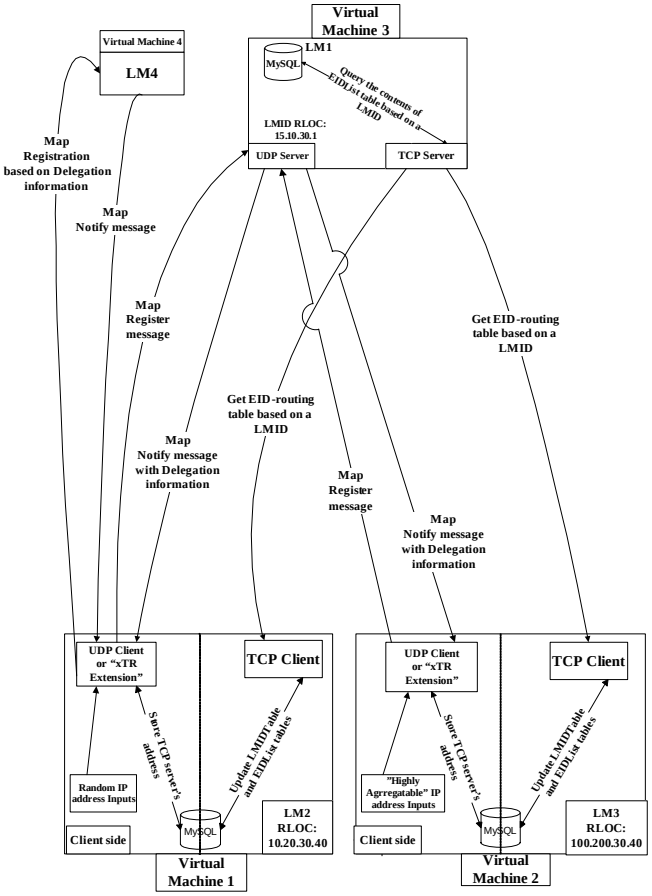

Fig. 2. Experimental topology for testing the capabilities of CRM. [9]

VMs 1 and 2 houses UDP client (i.e. "xTR Extension"), TCP client and a MySQL database. The dashed diving lines in the middle of VM 1 and 2 signify that, these VMs house elements of both client side (i.e. UDP client) and part of the LM (i.e. TCP client). However, the MySQL DB is common to both client and LM (that's why it is in the middle). In order to test multi-client handling capabilities, VMs 1 and 2 (as clients) are sending Map-Register messages to the LM1 (contained by VM3). VM 4 houses LM4 which is accessed by the UDP client (to send Map-Register message) in case of Delegation. The inner workings of LM1 and LM4 are same.

During delegation phase, the "current" LM will try to decide whether there is any other "better" suited LM for aggregation. In order to make that decision, the "current" LM needs to know what other LMs are advertising. Whenever there is a change in the network residing "behind" a LM, a new hash value is generated; which in turn, is advertised through BGPs community attribute. This phenomena prompts the "current" LM's TCP client to fetch the contents of *EID-routing table* belonging to the LM whose network state has changed; which in turn, enables each LM to know what other LMs are advertising (and thus overlay routing). Note that, in order to achieve minimal use of BGP advertisements, CRM through Quagga, only advertises community attributes; not the actual mappings.

Now, we are ready to input data for both LISP-ALT and CRM systems. A part of the input is from [10] and this sample was taken on 30/06/2010. This section of the source originates from randomized distribution of prefix lengths to mimic the real world. We have taken 100 such prefixes. As shown in figure 2 such inputs originated from VM1. The other part of the input is "highly aggregatable" and is generated from VM2 (see figure 2). We have used 21 such prefixes. Though the size of the input is small, we believe that, it is sufficient to demonstrate the capability of dynamic aggregation and provide a proof of concept.

The *"EID-routing"* table refers to a MySQL table that contains all the EID-prefixes aggregated and delegated by the current LM. Additionally, this table contains the orphan EID prefixes that the current LM will announce. A snapshot of a CRM's *"EID-routing table"* looks like the following:

LMID	EIDPrefix	EIDPrefixLength	Is_Delegated	Delegated_RLOC
15.10.30.1	110.0.1.0	24	Y	100.200.30.40
15.10.30.1	110.0.128.0	17	Y	100.200.30.40
15.10.30.1	110.0.16.0	20	Y	100.200.30.40
15.10.30.1	110.0.2.0	23	Y	100.200.30.40
15.10.30.1	110.0.32.0	19	Y	100.200.30.40
15.10.30.1	110.0.4.0	22	Y	100.200.30.40
15.10.30.1	110.0.64.0	18	Y	100.200.30.40
15.10.30.1	110.0.8.0	21	Y	100.200.30.40
15.10.30.1	110.1.0.0	16	Y	100.200.30.40
15.10.30.1	110.16.0.0	12	Y	100.200.30.40
15.10.30.1	110.2.0.0	16	Y	100.200.30.40
15.10.30.1	120.14.64.0	18	Y	10.20.30.40
15.10.30.1	63.130.121.0	24	N	X
15.10.30.1	81.130.29.0	24	N	X

Fig. 3. "EID-routing" table for CRM. [9]

In order to provide a viable comparison, we made a minimal implementation of LISP-ALT (to be accurate, we have realized the functionality of an ALT router) by observing the methods described in [6].As stated before, LISP-ALT only performs natural aggregation; as the first stage aggregation done in CRM. For this reason, the *"EID-routing"* table for LISP-ALT for the same set of inputs (i.e. same as CRM) will provide us with a different output snapshot:

LMID	EIDPrefix	EIDPrefixLength	Is_Delegated	Delegated_RLOC
15.10.30.1	110.0.1.0	24	N	X
15.10.30.1	110.0.128.0	17	N	X
15.10.30.1	110.0.16.0	20	N	X
15.10.30.1	110.0.2.0	23	N	X
15.10.30.1	110.0.32.0	19	N	X
15.10.30.1	110.0.4.0	22	N	X
15.10.30.1	110.0.64.0	18	N	X
15.10.30.1	110.0.8.0	21	N	X
15.10.30.1	110.1.0.0	16	N	X
15.10.30.1	110.16.0.0	12	N	X
15.10.30.1	110.2.0.0	16	N	X
15.10.30.1	120.14.64.0	18	N	X
15.10.30.1	63.130.121.0	24	N	X
15.10.30.1	81.130.29.0	24	N	X

Fig. 4. "EID-routing" table for LISP-ALT. [9]

It is evident from the tabular output that, LISP-ALT has no delegation capabilities. It can only perform natural aggregation and thus if the input is random (i.e. addresses that are not highly aggregatable) then LISP-ALT is forced to announce all the EID-prefixes through BGP. Therefore, the actual size of the RT becomes heavily dependent on the distribution of the prefixes.

The most important data structure for a mapping system dealing with distinct namespaces is, the *"EID-forwarding table"*. This MySQL table is derived/constructed from *"EID-routing table"* and delegation information available to the current LM. It houses all the locally owned registered and announced EID-prefixes. It also contains all the aggregates that are announced by the other LMs (i.e. delegation data when applicable). A sample snapshot of CRM's *"EID-forwarding table"* looks like the following:

LMID	EIDPrefix	EIDPrefixLength
15.10.30.1	63.130.121.0	24
15.10.30.1	81.130.29.0	24
10.20.30.40	120.0.0.0	8
100.200.30.40	110.0.0.0	8

Fig. 5. EID-forwarding table, generated by combining CRM's EID-routing table of 3 and delegation information. [9]

6. COST PROFILE, POSSIBLE OPTIMIZATIONS AND COMPARISONS

In order to evaluate CRM and compare it with LISP-ALT, we had to define and locate the functions that performed "key operations". Kcachegrind's call graph view aided us in this respect, by showing the frequentness of functions and their respective "inclusive costs" in percentages. The "inclusive cost" of a function is measured by the total number of CPU operations that occur between entering and exiting that function. Though the absolute number of CPU cycles might vary according to the underlying hardware, the percentages remain the same; making "inclusive cost" impervious to hardware change. The "inclusive cost" of all functions that are called (from the current function) are summed. That is why, for a generic C program, the function *main()* has a cost of *~100%*. However, "inclusive cost" is not the only criteria for defining whether a function performs any "key operation" or not. If a function executes some important logical procedures then it is also judged to have "key operations".

Before we go any further, we need to mention which parts of CRM and LISP-ALT are excluded from our current discussion. [5] We have refrained ourselves from using CRM's and LISP-ALT's UDP client side. Because, in the whole of things, UDP client's functionality bears miniscule cost compared to the UDP server side. Therefore, only the UDP server side will be used for the cost comparison between LISP-ALT and CRM. Additionally, we have excluded discussion about the TCP client-server portion. Because this part is a

[5]We have profiled every part of CRM and LISP-ALT in detail. This can be found in the thesis work [9]. This thesis has served as the basis of our current work.

novelty of CRM; LISP-ALT does not possess any such implementation.

As stated earlier, the UDP server is the part where we can observer the clear differences between LISP-ALT and CRM. At first, we examine the call graph view of LISP-ALT.

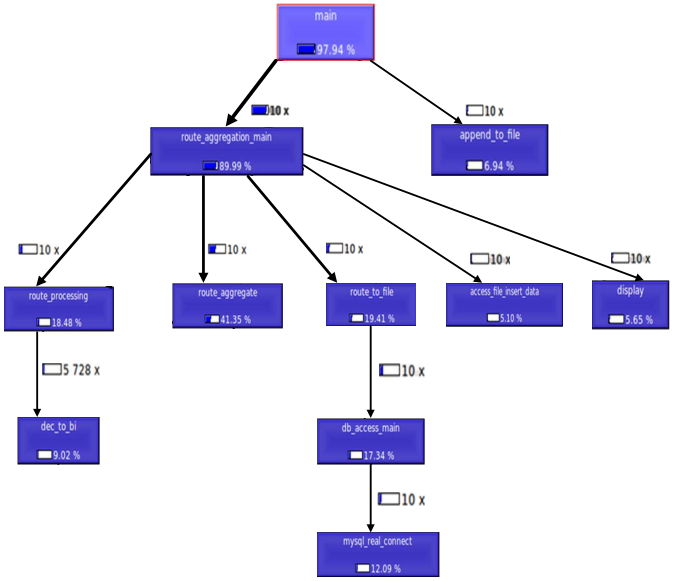

Fig. 6. Call graph view of LISP-ALT (formatted and summarized). [9]

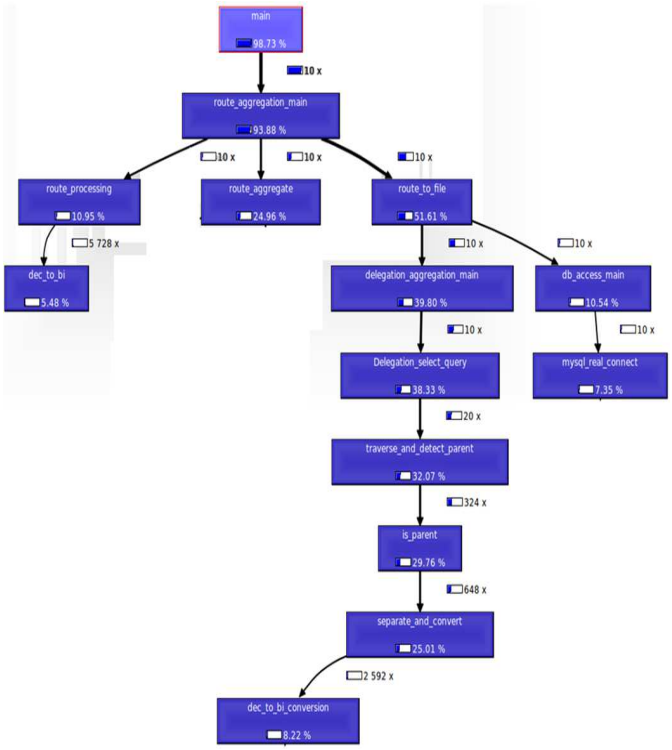

Fig. 7. Call graph view of CRM (formatted and summarized). [9]

It's no surprise when we see the *route_aggregate()* function, responsible for the natural aggregation, consumes ~42% of the total cost. Clearly, this function performs "key operations". The *route_processing()* function, in spite of its low cost (*~19%*), fills data structure holding the routes with necessary information required for natural aggregation. Hence, it is judged to have "key operations". The *route_to_file()* is also a function with "key operations"; because it seeks out the aggregated addresses and then puts it into the DB. Notably, the UDP server is not entirely dependent on the interactions with the DB (actually, this part of the program is computation centric) and thus we see a relatively low cost (*~18%*) of the *db_access_main()* function. Additionally we observe that, though the *dec_to_bi()* function is called more than 5,700 times, it is not a function with "key operations". In addition to its low cost, the *dec_to_bi()* function performs the trivial task of converting decimal numbers into binary. Therefore, just based on its frequency we cannot judge the *dec_to_bi()* function to possess "key operations".

In future optimizations, we should use a Radix tree [6] as the data structure that holds the addresses while aggregation is taking place (for both LISP-ALT and CRM), instead of linked list for brute-force ancestry detection ($O(n^2)$).

From prior discussion, we know that, in addition to the natural aggregation, CRM performs delegation and finally if needed generates Virtual Prefix. Our scheme for Virtual Prefix generation follows the algorithm of [4] which dictates that, if the size of the system does not grow quickly then Virtual Prefix is not created. As our system is quite small the Virtual Prefix generation part never gets executed and is thus absent from the call graph view.

[6]Radix Tree is successfully used in OpenLISP.

As mentioned previously, the first natural aggregation is common to both LISP-ALT and CRM. That is why, if we examine figures 6 and 7, we will find couple of common functions. These are: *route_aggregate()*, *route_processing()*, *route_to_file()*, *dec_to_bi()*, *db_access_main()* and *mysql_real_connect()*. These functions are essential for the 1st level aggregation. The comparison has to be made from the call graph of CRM (i.e. figure 7); because it houses all the functions (of both LISP-ALT and CRM). After close inspection of fig. 7, we calculate that, the functions required for 1st level aggregation costs *~50%*. With this, we can infer that, for one router, LISP-ALT costs about half of CRM. For the sake simplicity, we have changed the unit from percentage to cycles; i.e. we have assumed a cost of *~100%* is equal to 100 cycles. Therefore, when a CRM system of one router costs 100 cycles, a LISP-ALT system of one ALT router will cost 50 cycles. The following bar graph of figure 8 shows how the cost increases when the number of routers increase.
Looking closely at figure 8, it is apparent that, LISP-ALTs cost grows linearly. Because, for each router added (into the network), LISP-ALTs cost increases by 50 cycles. In terms of equation, this linear growth looks like:

$$Cycles = f(NumberOfRouters) = 50*(NumberOfRouters)$$

On the other hand, CRM's cost "enjoys" logarithmic growth. Because, CRM's cost can be described as a logarithm function of the number of routers;

$$Cycles = Constant * \log(Numberofrouters)$$

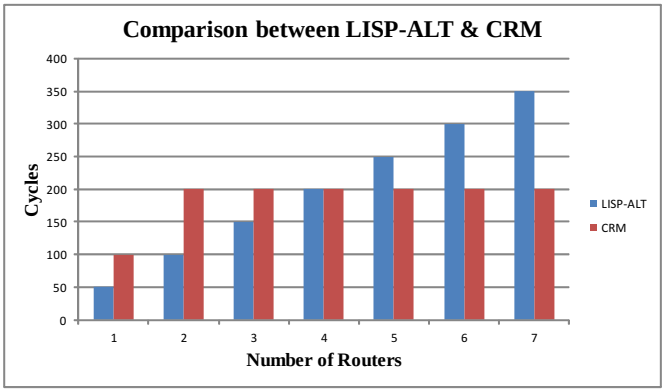

Fig. 8. Performance comparison between LISP-ALT and CRM. [9]

We can say the same about the amount of state within the system. Because, this is what compact routing is doing; it is compacting the routing state.

7. IMPLEMENTATION ANALYSIS

After the code profiling, it became apparent that, choosing MySQL for storing the network's "state" information was a costly choice (not to be confused with a bad/inappropriate choice). The APIs (e.g. *mysql_init()*, *mysql_real_connect()* etc.) that handled the communication between CRM and database has proven to be overwhelmingly expensive. We could have utilized something simpler e.g. CSV (Comma-Separated Values) files for storing the data. The data used to store the network's current state is not complicated enough to require the versatile functionalities of MySQL. Most of its functionalities were wasted. On the other hand, MySQL gave us the capability to normalize data; we were able to maintain uniqueness and consistency by using primary and foreign keys respectively. Though CSV files might have been cheaper; we would have been forced to use semaphores for handling concurrency issues; making it inherently slow.

8. CONCLUSION AND FUTURE WORK

The results and subsequent analysis shows that, CRM is feasible for deployment in the current Internet. Furthermore, our comparison has proven CRM to be far cheaper than LISP-ALT in terms of complexity.

At the moment, CRM is running on a handful of VMs. Therefore, in the future, it is pertinent that we test CRM in a more realistic network topology that simulates the genuine behavior of the Internet. Furthermore, we are now exchanging the mappings between LMs by establishing TCP connections. We predicted that, this will weigh down the network. At the time when CRM was being developed, Quagga did not have support for multiprotocol BGP that is used for MPLS; which in turn would have facilitated incremental distribution of routing table changes. CRM would have been benefited from such an approach. Ultimately, CRM would be integrated with OpenLISP; so that OpenLISP could run in the data plane whereas CRM would operate in the control plane.

References

[1] L. J. Cowen, *Compact routing with minimum stretch.*, Journal of Algorithms **38** (2001), 170 –183.

[2] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, *Locator/id separation protocol (lisp)*, Request for Comments, IETF, Internet Engineering Task Force, 2011.

[3] Dino Farinacci, Darrel Lewis, David Meyer, and Vince Fuller, *The locator/id separation protocol (lisp)* (2013).

[4] Hannu Flinck, Petteri Poyhonen, and Johanna Heinonen, *Analysis of landmark selection method*, 2011.

[5] V Fuller, D Farinacci, and D Meyer, *D. lewis," locator/id separation protocol alternative logical topology (lisp+ alt)*, RFC 6836, January, 2013.

[6] V. Fuller, D. Farinacci, D. Meyer, and D. Lewis, *Lisp alternative topology (lisp+alt)*, Request for Comments, IETF, Internet Engineering Task Force, 2011.

[7] V Fuller, D Lewis, and D Farinacci, *Lisp delegated database tree*, draft-fuller-lisp-ddt-01 (work in progress) (2012).

[8] V. Fuller and T. Li, *Classless inter-domain routing (cidr): The internet address assignment and aggregation plan*, Request for Comments, IETF, Internet Engineering Task Force, 2006.

[9] A. M. Anisul Huq, *Experimental implementation of a compact routing based mapping system for the locator/id separation protocol (lisp)*, Master's Thesis, 2012.

[10] G Huston, *As6447 bgp routing table analysis report*.

[11] Iannone, Saucez, and Bonaventure, *Implementing the locator/id separation protocol: Design and experience*, Computer Networks (2011).

[12] L. Iannone, D. Saucez, and O. Bonaventure, *Openlisp implementation report*, Internet-Draft, Network Working Group, 2009.

[13] Varun Khare, Dan Jen, Xin Zhao, Yaoqing Liu, Dan Massey, Lan Wang, Beichuan Zhang, and Lixia Zhang, *Evolution towards global routing scalability*, IEEE Journal on Selected Areas in Communications **28** (2010October), 1363 –1375. Institute of Computer Science,University of Wrzburg, Germany.

[14] Graham Mooney, *Evaluating compact routing algorithms on real-world networks*, Master's Thesis, 2010.

[15] Tapio Partti, *Improving internet inter-domain routing scalability*, Master's Thesis, 2011.

[16] Cristel Pelsser, Akeo Masuda, and Kohei Shiomoto, *Scalable support of interdomain routes in a single as*, Proceedings of the ieee global telecommunications conference, 2009, honolulu., 2009November, pp. 1 –8.

[17] Damien Saucez, Luigi Iannone, and Olivier Bonaventure, *Openlisp: An open source implementation of the locator/id separation protocol*, ACM SIGCOMM Demos Session (2009), 1–2.

[18] Damien Saucez, Luigi Iannone, Olivier Bonaventure, and Dino Farinacci, *Designing a deployable internet: The locator/identifier separation protocol*, Internet Computing, IEEE **16** (2012), no. 6, 14–21.

[19] Damien Saucez, Luigi Iannone, and Benoit Donnet, *A first measurement look at the deployment and evolution of thelocator/id separation protocol*, ACM SIGCOMM Computer Communication Review **43** (2013), no. 1, 37–43.

[20] M. Thorup and U. Zwick, *Compact routing schemes.*, Proceedings of the 13th annual acm symposium on parallel algorithms and architectures, 2001July, pp. 1 –10.